01A7
01A9 0A = LF
01AE 0D = CR

START = 0100

MP-R EPROM Programmer User's Guide

Introduction

The SWTPC EPROM Programmer and its accompanying software is a plug on option for the SWTPC 6800 Computer System. It is capable of programming any Intel 2716 (5 volt only) pin compatible EPROM. EPROM stands for Erasable Programmable Read Only Memory. Each 2716 is capable of storing 2K bytes of data and may be erased and reprogrammed up to one hundred times. The 2716 is erased by removing the opaque window cover from the top of the package and exposing the device to high level ultraviolet light for a specified amount of time. This amount of time will vary so consult the literature supplied with the ultraviolet light source that you plan to use. After erasing be sure to recover the glass window with the opaque cover. The 2716 can actually be erased by ambient sun and fluorescent light over a long period of time.

The SWTPC MP-A2 processor board on the 6800 Computer System has socket provisions for up to four 2716 pin compatible EPROM's. This gives the MP-A2 board the capability of up to 8K of EPROM. The EPROM's are switch addressable from CØØØ thru DFFF or from EØØØ thru FFFF. The CØØØ thru DFFF range is usually used for PROM BASIC or custom program applications. The EØØØ thru FFFF range is used for custom monitor or dedicated controller applications and may not be used simultaneously with the MP-A2's 6830 ROM monitor. The MP-A processor board has no provisions for EPROM memory.

EPROM Programmer Hardware and Software

The EPROM programmer hardware consists of a single, 5¼" x 5¼" circuit board which plugs onto one of the unused card slots on the interface bus of the SWTPC 6800 Computer System. The board contains a 24-pin integrated circuit socket where the EPROM to be programmed or verified is inserted. The EPROM may be programmed and verified from the socket on the programming board; however, the program code in the EPROM may not be executed from the EPROM programming board. You may duplicate EPROM's by plugging a programmed EPROM into the socket, having the programmer's software read and store the code, replace the programmed EPROM with an unprogrammed one and program using the stored code.

The software for the EPROM programmer consists of the programmer's monitor (distinct from the processor's monitor), editor and data table and occupies the lower 4K of memory. The monitor gives the user the ability to check, read, program and verify EPROM's. The editor gives the user the ability to load, modify and examine the data which will be stored to EPROM. Both the monitor and the editor occupy the first 2K bytes of memory when loaded into the SWTPC 6800 Computer System. The 2K bytes of data to be stored into the 2716 may be resident in any contiguous 2K byte memory segment. The programmer's software assumes the 2K to 4K segment if no specific addresses are given. This 2K byte block of memory wherever it may be located in memory is called the DATA TABLE. The starting address of this data table is called the BASE ADDRESS. When the EPROM programmer starts transferring code to the EPROM being programmed, the data stored in the BASE ADDRESS is stored in EPROM address ØØØØ. The data stored in the next sequential memory address is stored in EPROM address ØØØ1. The pattern continues all the way up to EPROM address Ø7FF, the highest address in the 2716 2K byte EPROM. Take note that the addresses within the EPROM's do not correspond with either the addresses of the data table or the actual addresses at which the EPROM will be accessed when installed on the MP-A2 board.

The data within the data table may be loaded by hand using the programmer's editor or may be loaded from a Mikbug® or SWTBUG® or SWTPC CORES assembler ASCII formatted object cassette or paper tape using the Programmer editor's load command. The object tape to be loaded must be contiguous data with only one ORG statement. Multiple ORG statements may only be used if they are followed only by RMB (reserve memory bytes) directives.

PC Board Assembly

NOTE: Since all of the holes on the PC board have been plated thru, it is only necessary to solder the components from the bottom side of the board. The plating provides the electrical connection from the

"BOTTOM" to the "TOP" foil of each hole. Unless otherwise noted it is important that none of the connections be soldered until **all** of the components of each group have been installed on the board. This makes it much easier to interchange components if a mistake is made during assembly. Be sure to use a low wattage iron (not a gun) with a small tip. Do not use acid core solder or any type of paste flux. We will not guarantee or repair any kit on which either product has been used. Use only the solder supplied with the kit or a 60/40 alloy resin core equivalent. Remember all of the connections are soldered on the bottom side of the board **only**. The plated-thru holes provide the electrical connection to the top foil.

( ) Before installing any parts on the circuit board, check both sides of the board over carefully for incomplete etching and foil "bridges" or "breaks". It is unlikely that you will find any; but should there be one especially on the "TOP" side of the board it will be very hard to locate and correct after all of the components have been installed on the board.

( ) Starting from one end of the circuit board install each of the three, 10-pin Molex female edge connectors along the lower edge of the board. These connectors must be inserted from the "TOP" side of the board and must be pressed down firmly against the circuit board so that each pin extends completely into the holes on the circuit board. Not being careful here will cause the board to either wobble and/or be crooked when plugging it onto the mother board. It is suggested that you solder only the two end pins of each of the three connectors until all have been installed at which time, if everything looks straight and rigid, you should solder the as yet unsoldered pins.

( ) Insert the small nylon indexing plug into the edge connector pin indicated by the small triangular arrow on the "BOTTOM" side of the circuit board. This prevents the board from being accidently plugged on incorrectly.

( ) Install all of the resistors on the circuit board. As with all other components unless noted, use the parts list and component layout drawing to locate each part and install from the "TOP" side of the board bending the leads along the "BOTTOM" side of the board and trimming so that 1/16" to 1/8" of wire remains. Solder. Make sure the extended leads of resistor R1 do not inadvertently contact any surrounding printed circuit traces.

( ) Install all of the capacitors on the circuit board. Be sure to orient electrolytic capacitors C2 and C5 as shown in the component layout drawing. Solder.

( ) Install the transistors and diodes. These components must be oriented to match the component layout drawing. Solder.

NOTE: Install transistor Q5 so that its metal base is spaced away from the circuit board about 1/16" to prevent the transistor's case from contacting the circuit board's foil. A special nylon spacer may be supplied with the kit for this purpose. If so, use it. Install diode D3 so that its flat side matches with that shown in the component layout drawing.

( ) Install integrated circuit IC2 on the circuit board. This component must be oriented so its metal face is facing the circuit board and is secured to the circuit board with a #4- 40 x 1/4" screw, lockwasher and nut. A heatsink is not used. The three leads of the integrated circuit must be bent down into each of their respective holes. Solder.

( ) Install integrated circuits IC3, IC4 and IC5 on the circuit board. Do not bend the leads on the back side of the board. Doing so makes it very difficult to remove the integrated circuits should replacement ever be necessary. The semi-circle notch or dot on the end of the package is used for orientation purposes and must match with the outlines shown on the component layout drawing for each of the IC's. Solder.

( ) Install inductor L1 on the board. Solder.

( ) Install the 24-pin integrated circuit socket for IC6 on the circuit board. Orient the socket so the

handle is adjacent to the top, right corner of the circuit board. Solder the socket with the contacts in the "OPEN" position.

NOTE: MOS integrated circuits are susceptible to damage by static electricity. Although some degree of protection is provided internally within the integrated circuits, their cost demands the utmost in care. Before opening and/or installing any MOS integrated circuits you should ground your body and all metallic tools coming into contact with the leads, thru a 1 M ohm ¼ watt resistor (supplied with the kit). The ground must be an "earth" ground such as a water pipe, and not the circuit board ground. As for the connection to your body, attach a clip lead to your watch or metal ID bracelet. Make absolutely sure you have the 1 Meg ohm resistor connected between you and the "earth" ground, otherwise you will be creating a dangerous shock hazard. Avoid touching the leads of the integrated circuits any more than necessary when installing them, even if you are grounded. On those MOS IC's being soldered in place, the tip of the soldering iron should be grounded as well (separately from your body ground) either with or without a 1 Meg ohm resistor. Most soldering irons having a three prong line cord plug already have a grounded tip. Static electricity should be an important consideration in cold, dry environments. It is less of a problem when it is warm and humid.

( ) Install MOS integrated circuit IC1 following the precautions given in the preceding section. As it is installed, make sure it is down firmly against the board before soldering all of the leads. **Do not** bend the leads on the back side of the board. Doing so makes it very difficult to remove the integrated circuit should replacement ever be necessary. The "dot" or "notch" on the end of the package is used for orientation purposes and **must** match with that shown on the component layout drawing for the IC. Solder.

( ) Working from the "TOP" side of the circuit board, fill in all of the feed-thru's with molten solder. The feed-thru's are those unused holes on the board whose internal plating connects the "TOP" and "BOTTOM" circuit connections. Filling these feed-thru's with molten solder guarantees the integrity of the connections and increases the current handling capability.

( ) Now that all of the components have been installed on the board, double check to make sure all have been installed correctly in their proper location.

( ) Check very carefully to make sure that all connections have been soldered. It is very easy to miss some connections when soldering which can really cause some hard to find problems later during checkout. Also look for solder "bridges" and "cold" solder joints which are another common problem.

Since the MP-R circuit board now contains MOS devices, it is susceptible to damage from severe static electrical sources. One should avoid handling the board any more than necessary and when you must, avoid touching or allowing anything to come into contact with any of the conductors on the board.

Parts List - MP-R EPROM Programmer

Resistors

——	R1	4.7K ohm 1/2 watt resistor	——	R7	4.7K ohm 1/4 watt resistor
——	R2	470 ohm 1/4 " "	——	R8	470 ohm " " "
——	R3	1 ohm 1/2 " "	——	R9	470 ohm " " "
——	R4	24.3K ohm precision resistor	——	R10	470 ohm " " "
——	R5	1.21K " "	——	R11	330 ohm " " "
——	R6	10K ohm 1/2 watt resistor			

Capacitors

——	C1	150 pfd capacitor	——	C4	0.1 mfd capacitor
——	C2	*470 mfd electrolytic capacitor	——	C5	*100 mfd electrolytic capacitor
——	C3	0.1 mfd capacitor			

Semiconductors

——	D1	*1N4003 silicon diode	——	Q2	*2N5210 NPN transistor
——	D2	*1N4003 silicon diode	——	Q3	*2N4402 PNP transistor
——	D3	*LED	——	Q4	*78L05 5 volt regulator
——	Q1	*2N5210 NPN transistor	——	Q5	*SS1122 PNP transistor

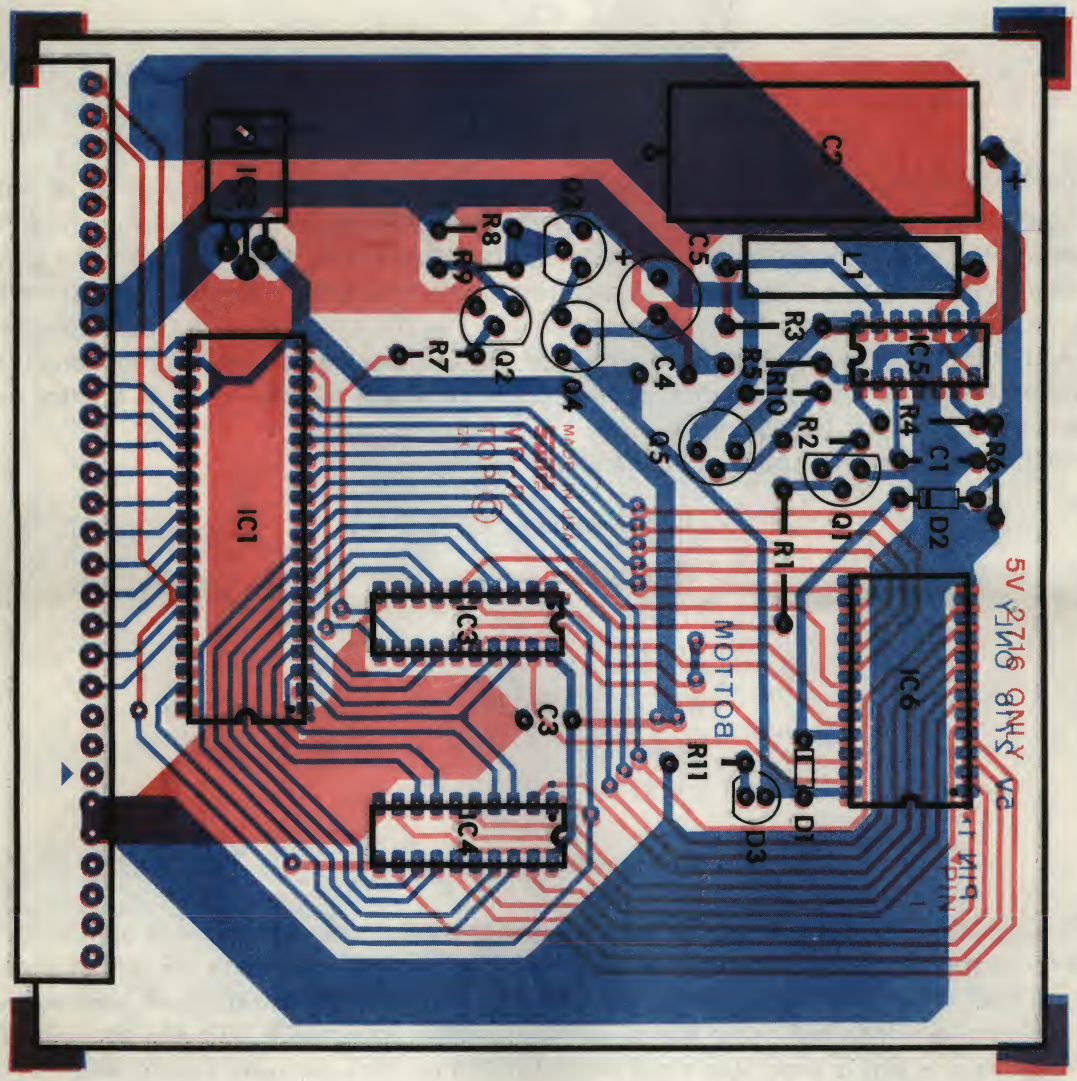
Integrated Circuits

——	IC1	*6820 or 6821 PIA	——	IC4	*74LS273 octal D-flop
——	IC2	*7805 5 volt voltage regulator	——	IC5	*TL497 switching regulator
——	IC3	*74LS273 octal D-flop	——	IC6	2716 (5volt only EPROM and not supplied with kit)

Miscellaneous

—— L1 200uH, 1A inductor

NOTE: All components flagged with a * must be orientated as shown in the component layout drawing


C2
L1
IC5
C5
C4
R8
R9
R7
Q2
Q4
Q5
R3
R10
R2
R4
C1
D2
R6
Q1
R1
IC1
IC3
C3
IC4
IC6
D1
D3
R11
MOTTOB
5V ONLY
VCC
PIN 1

Using the Programmer

The EPROM programmer may be plugged onto any one of the unused I/O card positions on the back of the SWTPC 6800 Computer System. Upon executing the programmer's software the computer will respond with a request for the I/O port number onto which the EPROM programmer board is plugged. The correct I/O number should be entered at that time. Don't forget that the I/O's are numbered from Ø thru 7 left to right. The correct number for each I/O position is printed on the back of the mother board adjacent the interface.

```
SWT PRMPRG VER 1.0


I/O PORT #3
ARE YOU SURE? Y
```

EPROM Programmer's Monitor

Once the EPROM programmer software is loaded into the system via cassette or paper tape and the program is started by typing G for "Go to the User Program", the system comes up in the EPROM programmer's monitor which is distinct from the system's operating system ROM monitor. The programmer's monitor prompts with a > and is capable of responding with any one of seven two letter commands. You need only type the first two characters of each command. The computer prints out the remainder for you. The BASE ADDRESS is initially set at ØBØØ and must be reset before the WRITE command if you desire something other than this. Typing a "Control X" at any time will transfer program control back to the monitor. There is no provision for backspace.

The EPROM programmer monitor's commands are as follows:

UNPGMMED CHECK
READ
PRINT
EDIT
WRITE TO PROM
VERIFY
EXIT

Command Summary

UNPGMMED CHECK—The unprogrammed check command verifies that every byte in the EPROM over the address range specified is unprogrammed (=FF). If any bytes are not set to FF as they should be, the user is notified, otherwise the message UNPGMMED is printed. Be sure to check each erased EPROM before you program it to make sure that all bytes have been reset to FF. The address range over which the EPROM is checked is selectable since the EPROM may be selectively programmed. It is possible to program any segment of the EPROM without disturbing non-selected addresses. Example:

```
>UNPGMMED CHECK
BGN PROM ADDR=0000
END PROM ADDR=07FF
UNPGMMED
```

or

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED
```

or

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
```

```
*ERROR*
INCON: 0000 DATA=FF PROM=00
CONTINUE? Y

*ERROR*
INCON: 0001 DATA=FF PROM=00
CONTINUE? N

>
```

In this command as with all others, responding with a carriage return to the "BGN ADDR=" question will default the programmer to the entire EPROM address range (ØØØØ thru Ø7FF). Responding with a carriage return to the "END ADDR=" question will default the programmer to the single byte entered for the "BGN ADDR=" question.

READ—The READ command is used to transfer information from an already programmed EPROM into the data table so that either part or all of the data may be edited and/or written into another EPROM. The READ command automatically resets the BASE ADDRESS to Ø8ØØ and asks for the address range to be transferred from the EPROM being read to the DATA TABLE. Executing a READ with no EPROM installed in the socket will fill the DATA TABLE with ØØ's over the address range specified.

```
>READ
BGN PROM ADDR=<CR>   CARRIAGE RETURN

>
```

PRINT—The PRINT command prints the DATA TABLE over the EPROM address range specified. The data is printed with the EPROM address followed by sixteen bytes of data, repeated until the entire specified address range has been printed. You may prematurely exit the PRINT sequence by repeatedly striking any key until a ">" is printed. Example:

```
>PRINT
BGN ADDR=0000
END ADDR=001F

<ADDR> <DATA>

0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

>
```

EDIT—The EDIT command transfers control to the programmer's editor which is described later in this manual. Example:

```
>EDIT

TYPE B,M,L, OR X:

>
```

WRITE TO EPROM—The WRITE TO EPROM command transfers the data stored in the DATA TABLE to the EPROM over the EPROM address range specified. Upon completion the EPROM is automatically verified for accuracy. If a discrepancy is found the program prints BYTE WILL NOT PROGRAM followed by the location of the discrepancy. The total program and verification time varies up to two minutes. **Do not** RESET or power down while writing to the EPROM. Example:

```
>WRITE TO PROM
BGN PROM ADDR=0000
END PROM ADDR=0100
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD
```

or

```
>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING

*ERROR*
BYTE WILL NOT PGM
0000 DATA=FF PROM=00
CONTINUE? Y

*ERROR*
BYTE WILL NOT PGM
0001 DATA=FF PROM=00
CONTINUE? N

>
```

VERIFY—The VERIFY command verifies that the data stored in the EPROM over the address range specified is the same as the data stored in the DATA TABLE. This function is automatically executed as part of the WRITE command. If no discrepancies are found, the message VERIFIED GOOD is printed. Example:

```
>VERIFY
BGN PROM ADDR=(CR)
VERIFIED GOOD

>
```

or

```
>VERIFY
BGN PROM ADDR=0000
END PROM ADDR=0100

*ERROR*
INCON: 0000 DATA=FF PROM=00
CONTINUE? Y

*ERROR*
INCON: 0001 DATA=FF PROM=00
CONTINUE? N

>
```

EXIT—The EXIT command returns control to the MIKBUG® or SWTBUG® monitor. PRMPRG may be reentered, however, without losing data stored in the DATA TABLE simply by typing G for Go to User Program. Exiting and then restarting the program will reset the BASE ADDRESS to the default value of Ø8ØØ. Example:

```
>EXIT

$
```

EPROM Programmer's Editor

When the command EDIT is typed when in the EPROM programmer's monitor, control is transferred to the EPROM programmer's editor. From the Editor, the user may modify or create a DATA TABLE any-

where within computer memory except for the lower 2K bytes (ØØØØ - Ø7FF) where the EPROM Programmer software resides. Once the Editor is entered, the following message is typed on the screen:

TYPE B, M, L or X:

These characters are defined below:

B —change BASE ADDRESS of the DATA TABLE

M —modify the DATA TABLE (change data or create DATA TABLE)

L —Read object file from assembler or monitor punch routine

X —exit Editor and return to programming monitor

These command characters are defined as follows:

B —When **B** is typed, the old BASE ADDRESS of the DATA TABLE is printed. The user may keep that address by typing a carriage return, or may change it by typing a 4-digit hex address which will become the new BASE ADDRESS.

M —When the **M** command is typed, the computer will respond with:

NEW TABLE?

If a **Y** for yes is entered the BASE ADDRESS is set to the default value of Ø8ØØ and all data within the table is set to $FF. If a **N** for no is entered, the BASE ADDRESS is left at whatever value it was previously set to. If it was not previously set by the user, it will be at the default value of Ø8ØØ. The computer will then respond with:

BGN PROM ADDR=

You should at this time type in the 4-digit hex address of the EPROM memory address you wish to examine and/or change. The address may be from ØØØØ thru Ø7FF inclusive. Attempting to access an address out of this range will return you back to the editor. The computer will line feed, return, echo the address, display its contents and a space.
At this point the user has the option of advancing, either forward or backward, to the next memory location, or changing the data stored at the displayed address and advancing to the next location or of exiting the **M** function.

To display the next sequential address and data, type a line feed. Any leading spaces that are entered will be ignored by the memory change function.

To display the next sequential address going backward from the present location, a ↑ should be entered.

To change the data stored at the displayed location, enter the new data, either with or without a leading space. If a non-hex value, such as a 3Q is entered the data will remain unchanged a "?" will be printed and the address will be repeated. If the data is unable to be changed (write protected memory, etc.) a ? will be output and the address will be repeated.

To exit the Memory Examine and Change function, type a carriage return. Example:

```
TYPE B,M,L, OR X: M
NEW TABLE? N
BGN PROM ADDR=0000
0000 FF 01
0001 FF  02
0002 FF   03
0003 FF    04
0004 FF ^
0003 04 ^
0002 03 ^
0001 02 ^
0000 01 (CR)
TYPE B,M,L, OR X:
```

L —The L or LOAD command is used to load a SWTPC CORES editor/assembler, MIKBUG® or SWTBUG® ASCII formatted tape into the DATA TABLE. The BASE ADDRESS of the DATA TABLE is automatically reset to the default address of Ø8ØØ and all object data is loaded in sequentially from this address regardless of the assigned object address. For example, an assembler object tape ORG'ed to be located sequentially from Ø1ØØ thru Ø3ØØ inclusive would be loaded into the data table from Ø8ØØ thru ØAØØ inclusive. Since programs to be stored to ROM may only be located from CØØØ thru FFFF on the MP-A2 board, this Ø1ØØ thru Ø3ØØ program would have to be written with address independent code (no extended addressing) to actually function at an address location other than which it was ORG'ed at. Typical EPROM programs will have to be ORG'ed at low addresses so that they may be written and debugged in RAM memory. Once debugged, the program may be ORG'ed and reassembled for high memory where it may be burned into EPROM. If the anticipated EPROM program is to be stored in the CØØØ thru DFFF region, extra MP-M or MP-8M memory boards may be modified as per the MP-A2 instructions to be addressed in high memory so that programs may initially be assembled and tested in high memory from the start without the need to go back and re-ORG and reassemble.

Since the LOAD command loads all data sequentially starting from the BASE ADDRESS, only one ORG statement is allowed in assembler generated code unless subsequent ORG statements are followed by RMB (reserve memory byte) directives only. All object data is stored sequentially from the Ø8ØØ BASE ADDRESS. If you are trying to store an 8K byte program into four sequential 2K byte EPROM's. The object code data for these EPROM's would be stored sequentially from memory location Ø8ØØ thru 27FF. This is a total of 10K bytes of RAM memory. To program these EPROM's we would first erase (if not already so) the four EPROM's we intend to use. We would then insert the EPROM programming board into one of the unused card positions on the back of the SWTPC 6800 Computer System and load and execute the PRMPRG program as follows:

```
                    PRMPRG VER 1.0

                I/O PORT #3
                ARE YOU SURE? Y
```

We then enter the programmer's editor and use the L command to load the 8K byte object tape. When we enter the L, the computer responds with the BLOCK #. If we have enough memory to store all of the object code to be stored to EPROM we enter 0. This causes the computer to load object data sequentially from the very first byte on the tape. If we have less than enough memory, say for example only 4K (2K of which is for the PRMPRG program itself) we must load the object data to memory, 2K blocks at a time. Specifying BLOCK #1 loads the second 2K block, and so on; all the way up to block #9. The disadvantage here is that the object tape must be repeatedly loaded to reposition the data in the DATA TABLE. In the following example let's assume that we have 10K of memory so that all of the object code may be read and stored in one pass-

```
>EDIT

TYPE B,M,L, OR X: L
BLOCK #0                    Insert object tape in recorder and PLAY.
TYPE B,M,L, OR X: X         Read data will be printed on the terminal.
                            Insert the first EPROM in the programmer's socket
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED
```

Continued next page

```
>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: B
BASE ADDR=0800  1000
TYPE B,M,L, OR X: X

>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: B
BASE ADDR=1000  1800
TYPE B,M,L, OR X: X

>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: B
BASE ADDR=1800  2000
TYPE B,M,L, OR X: X

>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>
```

Reset BASE ADDRESS for the next 2K block

Remove the first and
Insert the second EPROM in the programmer's socket

Reset BASE ADDRESS for the next 2K block

Remove the second and
Insert the third EPROM in the Programmer's socket

Reset BASE ADDRESS for the next 2K block

Remove the third and
Insert the fourth EPROM in the programmer's socket

Remove the fourth EPROM

In the following example, we load the same program one block at a time as if we have only 4K of memory in our computer system:

```
>EDIT

TYPE B,M,L, OR X: L
BLOCK #0
TYPE B,M,L, OR X: X
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED
```

Insert object tape in recorder and PLAY.
Read data will be printed on the terminal
Insert the first EPROM in the programmer's socket

```
>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: L
BLOCK #1
TYPE B,M,L, OR X: X
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED
```

Rewind and re-insert the object tape and PLAY

Remove the first and insert the second EPROM in the programmer's socket

```
>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: L
BLOCK #2
TYPE B,M,L, OR X: X
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD
```

Rewind and re-insert the object tape and PLAY

Remove the second and insert the third EPROM in the programmer's socket

Continued next page

```
>EDIT

TYPE B,M,L, OR X: L
BLOCK #3                    Rewind and re-insert the object tape and PLAY
TYPE B,M,L, OR X: X

>UNPGMMED CHECK             Remove the third and insert the fourth EPROM
BGN PROM ADDR=(CR)          in the programmer's socket
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>                           Remove the fourth EPROM
```

X —The X command is used to EXIT the Editor and return to the programmer's monitor. Example:

```
>EDIT

TYPE B,M,L, OR X: X

>
```

A Word of Warning

The timing and voltage switching for the MP-R EPROM Programmer are under software control. The control portion of the program resides in the lower 2K bytes of memory and it is very important that this memory segment as well as the rest of the processor operate flawlessly. If there is a problem it is possible to permanently damage the EPROM being programmed. It is suggested therefore that you run the memory diagnositics on your system prior to using the EPROM programmer just as a precaution. It is also suggested that you familiarize yourself with the operation of the programmer's software by operating the board without a EPROM installed until you feel comfortable with the software. Do not RESET or power down the system while you are WRITING to EPROM.

Testing the Programmer

It is very important that you check to make sure that the 25.0 VDC programming voltage is correct before attempting to program an EPROM. Clip the leads of a DC voltmeter across electrolytic capacitor C2. Use the polarity markings of the capacitor for proper connection. Load the PRMPRG software and give it a READ command over the entire address range with no EPROM installed in the socket. This will fill the data table with zeros (ØØ's). Now execute a WRITE command over the entire address range. The voltmeter

should read between 24.0 and 26.0 VDC. Don't forget to take into account the inaccuracies in your voltmeter. If the measured voltage is too high, you should increase the value of resistor R5 approximately 5 ohms for each 0.1 volt difference. If the voltage is too low, you should increase the value of resistor R4 approximately 100 ohms for each 0.1 volt difference. The ideal voltage is 25.0 VDC. Having the voltage too high (above 26.0 volts) can destroy the EPROM being programmed. Recheck the voltage after making any changes.

How It Works

Interfacing from the 6800 bus to the EPROM and its programming circuitry is done thru PIA IC1. Since more data lines than were available on one PIA are necessary, latches IC3 and IC4 are used to provide the extra outputs. Data is loaded into the latches by strobing the clock line of the latches with the CA2 and CB2 line of the PIA. Since no DC voltage higher than 12 volts is available on the SWTPC 6800 Computer System bus, it is necessary to use an integrated swtiching regulator to up the unregulated 7-8VDC bus voltage to the 25.0 VDC required for programming. Transistors Q1 and Q5 control the voltage to the switching regulator so the switcher may be disabled when not in use. Since the EPROM is usually installed and removed while computer system power is applied, transistors Q2 and Q3 control the EPROM +5 VDC regulator so that power is not applied to the socket during installation and removal.

The programming sequence is under total software control and starts by setting the CS input high which is the normal state during programming. The PD/PGM input is set low and the 25.0 VDC power supply is switched on and applied to the Vpp input. The selected address and data to be stored are applied to address and data inputs of the EPROM and the PD/PGM input is pulsed high for 45 to 55 milliseconds which programs the byte. This process is repeated for all bytes in which at least one bit must be set to zero. Only those bytes which must be set to something other than FF's are actually programmed. The unprogrammed (erase) state of each byte in the 2716 EPROM is FF.

The read sequence sets the PD/PGM input and CS inputs low, sets the selected byte address up on the EPROM, and inputs the EPROM data thru the B side of the PIA.

Handling the 2716 EPROM

The 2716 EPROM is a MOS device and is susceptable to damage from static electricity. It is shipped and should be stored with its leads impressed into a conductive material (usually conductive foam) for maximum protection. To transfer the 2716 from the conductive foam to the programmer's socket, place one hand on the conductive foam and grasp all of the leads of the IC with the other hand. Carefully remove the IC. While grasping all of the leads with the fingers of one hand grab the computer chassis with the other. This brings both to the same voltage potential level. While contacting the computer's chassis install the IC into the programmer's zero force socket. Make sure, of course, that the socket is open and ready to accept the IC and that the IC is oriented correctly. The dot or notch on the end of the IC's package must match with the PIN 1 indicator on the PC board. After insertion, double check for proper orientation and close the latching socket.

The MP-R's LED diode D3 is lit whenever +5 VDC power is applied to the socket. **Never** install or remove the EPROM to the socket while this diode is lit. The light is normally off unless data within the EPROM is being accessed.

After programming, follow the same precautions when transferring the 2716 to the conductive foam or MP-A2 board. Place one hand on the computer's chassis and open the socket. While contacting the computer's chassis, remove the IC with the fingers in contact with as many of the IC's pins as possible. While grasping the IC with one hand place the other on either the conductive foam or MP-A2 board and carefully insert the IC. If you are inserting the IC into the MP-A2 board, be sure to orient the IC as indicated on the MP-A2 board or its component layout drawing. Remember also that the lid of the 2716 should be covered after programming to prevent ambient light from erasing the IC over a long period of time.

Erasing 2716 EPROM's

To erase the 2716 EPROM it must be exposed to an ultraviolet light source emiting wavelengths shorter than 4000 Angstroms. Although fluorescent lights and sunlight emit wavelengths in this region, the amount of time required for erasure would vary from months to years. Those special UV sources emitting wavelengths required for erasing are hazardous to the skin and eyes and must be handled with care. Typical erasing time using a short wavelength UV lamp varies from twenty to sixty minutes. EPROM's should not be erased while power is applied since current paths exist that effectively cancel the energy being provided by the UV light.

At the time of this writing only one manufacturer to our knowledge has been advertising an EPROM eraser for the hobbyist. This is:

Ultra-Violet Products Inc.
5100 Walnut Grove Ave.
San Gabriel, CA 91778

The erasers are sold thru dealers and retail for $59.50. If you would rather build your own there is an article on just that on page 91 of the January 1977 issue of BYTE magazine.

Error Messages

When either user errors or execution errors occur, the user is notified thru error messages. Some errors require user response to continue execution. For example, if an inconsistancy is found using the verify routine, the following message is printed:

```
*ERROR*
INCON: 00F6 DATA=C1 PROM=D1
CONTINUE?
```

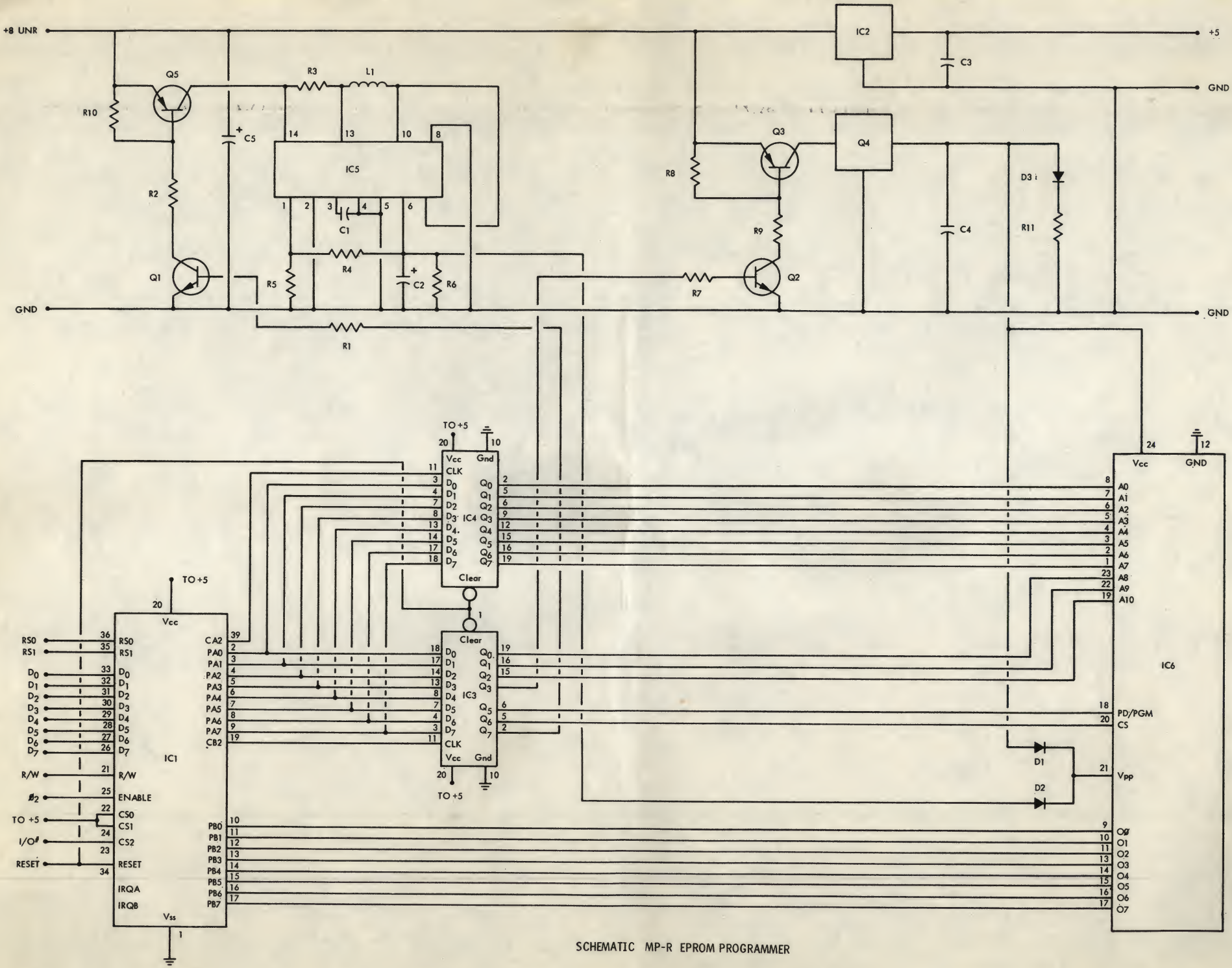
The user may acknowledge the error and continue by typing a **Y** for yes or may abort and return to the monitor by typing a **N** for no. The following error messages may be generated:

OUT OF RANGE—The specified address limits are larger than Ø7FF, which is the capacity of the 2716 EPROM.

BYTE WILL NOT PGM—The address in the EPROM printed after this message will not program correctly.

END<BGN—Specified addresses are out of order.

INCON:– A data inconsistency has been found.



SCHEMATIC MP-R EPROM PROGRAMMER

SWTPC 6800 DISASSEMBLER

BY MICHAEL J. DENNIS

The DESEMBLER program can be used to "de-assemble" a loaded machine code program. This may be desirable to aid debugging or to assist in understanding, correcting or modifying a machine code supplied program. The DESEMBLER lists the memory address of each byte of the instruction, the equivalent ASCII character of each byte, the assembly language mnumonic of the machine code instruction along with addressing symbols and the operand. Some of the features of this program are:

The DESEMBLER can be run from any location in memory.

Program has a built in "move" routine which will move the DESEMBLER to the desired memory location.

Program has a "byte search" option which will find every occurance of a specified byte within a memory area and print it out with its address, the preceeding byte, and the following two bytes.

The DESEMBLER has an optional "intersection mark" feature which marks in the desembled listing the program locations to which the program branches or jumps to. These marks indicate whether it was a direct branch or a subrountine type branch. This option will also cause a symbols table listing of the intersection addresses to be printed after the regular desembler listing.

Output may be directed to any port via a control, asynchronous (ACIA), or parallel (PIA) interface.

No memory locations need be initialized prior to running the DESEMBLER other than the program counter (A048-A049).

Hardware configuration and software support:

This program is designed to run in any 2.5K of continous memory on a SWTPC 6800 computer system. Program requires MIKBUG* software and much of the MIKBUG* RAM memory (A000-A07F). The DESEMBLER communicates with the control terminal at port #1 and asynchrounos serial output devices such as a TELETYPE* or parallel output devices such as a PR-40 printer are supported as options at any other port. The DESEMBLER as supplied initially loads into memory locations 0100-0AFF.

*MIKBUG is a registered trademark of Motorola Inc.
*TELETYPE is a registered trademark of Teletype Inc.

Operation of the DESEMBLER:

Upon loading A048 and A049 with the starting address (0100) and typing the MIKBUG* command "G" to start, the DESEMBLER will identify itself on the control terminal and ask for the option wanted as shown.

SWTPC DESEMBLER V.4

OPTION (D,S,M,E):

The user should then type a D, S, M, or E where:

D indicates a desembled listing is desired. (disassembly)
S for byte search routine.
M for moving DESEMBLER program to new memory location.
E for exiting back to MIKBUG*.

If "E" is entered, the program immediately returns to MIKBUG*. To restart DESEMBLER type MIKBUG* command "G".

If "M" is typed the program will ask:

NEW BEGINNING ADDRESS:

The user should key in the desired 4 digit hex address that the program should be moved to. The program will restart upon completion at the new location. When a move is made, be sure that sufficient memory exists for the program and for the creation of the intersection data file (symbol table) if intersection marks are to be printed. Intersections point to places in the program to where a branch or jump instruction terminates.

If the "D" option is chosen, the DESEMBLER will ask:

MARK INTERSECTIONS? (Y/N)

If the user wants the program to indicate the address locations which the desembled program branches or jumps to, he should type a "Y" for yes, otherwise "N" should be typed for no. In the listing, locations that are referred to by a BSR or JSR instruction will be flagged by a >. Locations referred to by a branch instruction are flagged by a -. The DESEMBLER will then ask:

INPUT FILE ADDRESS:

The user should key in the 4 digit hex beginning address of the area of memory to be desembled. The program will then type "TO" and the user should key in the ending address of this memory area. The DESEMBLER will than ask:

PORT & DEVICE (0-7; C,S,P):

The user should respond with the port number and the output device type where "C" is the control interface, "S" is a serial (ACIA) interfaced device, and "P" indicates a parallel interfaced printer. Note that if "C" is chosen, output will be to port #1 regardless of the entered port number.

The DESEMBLER will then type:

HEADER:

The user should respond with the title to be printed on the desembled listing, up to 44 characters in length, terminated by a carriage return.

If "S" was the option chosen, indicating a "byte search", the program will ask for the input file address and the port and device as it would for the "D" option. The program will then ask for the "BYTE DESIRED:". The user should enter the 2 digit hex number. The DESEMBLER will then output to the selected output device the address of each occurance, the preceeding byte, the byte of interest (corresponding to

the address), and the following two bytes. To exit from this routine a reset of carriage return must be hit. To restart program type the MIKBUG* command "G".

Input notes:

If an illegal option or output device is entered, the DESEMBLER will re-ask the question. If an error occurs in the input of a number, however, program control returns to MIKBUG*. The command "G" will restart the DESEMBLER. If too many characters are entered in the header title, the carriage return will be assumed as the 44th character and output will begin.

Output notes:

The "desemble" option of the DESEMBLER will produce a pagenated output including cut marks, the header title, and the page number. The page will list 56 (38 hex) lines which include the memory address of the instruction, the memory contents and ASCII equivalent of each byte in the instruction, the instruction mneumonic, addressing symbols and operand. The addressing symbols and operand are in the following format:

blank			for inherent addressing
# 3E	or	# AØ3E	for immediate addressing
AØ3E	or	ØØ3E	for extended or direct addressing
X, 3E			for indexed addressing
Ø23E			for relative addressing

Note that all numbers are hexadecimal. Also note that relative addressing gives the absolute address of the branch rather than the relative address contained in the next memory location.

Next to the memory contents in the DESEMBLER output is printed the ASCII equivalent of the byte or a space. A space is printed for all control characters or non-capital ASCII characters. Whether or not the equivalent ASCII character is printed out for other values depends on the parity of the memory contents (the most significant bit) and on the instruction at the relative address of Ø286 (hex) in the DESEMBLER program. If the starting address of the DESEMBLER + 286 (hex) contains

- 2Ø then no ASCII characters will be printed,
- 2B then ASCII characters with parity=Ø will be printed (default),
- 2A then ASCII characters with pari-y=1 will be printed,
- 25 then all ASCII characters will be printed regardless of parity.

The number of lines printed per page is controlled by the number in the relative address 246 of the DESEMBLER program (normally 38 hex). (Ø286 relative is location Ø386 and Ø246 relative is Ø346 if the DESEMBLER is origined at Ø1ØØ as supplied. When a move is done the locations will naturally change.)

Below is a sample run of the DESEMBLER with some comments on the output.

NOTE Location ØA14 in the object dump should be 42 and location ØA15 should be C9.

```
*G SWTP 6800 DESEMBLER V.4

OPTION (D,S,M,E): D
MARK INTERSECTIONS? (Y/N) Y

INPUT FILE ADDRESS: 0100 TO 0175
PORT & DEVICE (0-7; C,S,P): 7P

HEADER: DESEMBLER VER. 4.0

====


DESEMBLER VER  4 0          PAGE 01


0100 34         4                DES
0101 34         4                DES
0102 20 6B                       BRA     : 016F
0104 FE A048     H           -   LDX       A048
0107 FF A014                     STX       A014
010A 86 71                       LDA A #   71
010C 08                      -   INX
010D 4A         J                DEC A
010E 26 FC      &                BNE     : 010C
0110 FF A016                     STX       A016
0113 CE A054     T               LDX     # A054
0116 FF A04E     N           -   STX       A04E
0119 FE A016                     LDX       A016
011C A6 00                       LDA A X.    00
011E 08                          INX
011F FF A016                     STX       A016
0122 FE A04E     N               LDX       A04E
0125 A7 00                       STA A X     00
0127 08                          INX
0128 81 3B      ,                CMP A #   3B
012A 26 EA      &                BNE     : 0116
012C 86 06                       LDA A #   06
012E 8D 67                       BSR     : 0197
0130 BD E047     G               JSR       E047
0133 FF A04A     J               STX       A04A
0136 FF A04C     L               STX       A04C
0139 FF A048     H               STX       A048
013C 5F         +                CLR B
013D B6 A04A     J               LDA A     A04A
0140 B1 A014                     CMP A     A014
0143 2B 0A      +                BMI     : 014F
0145 2E 13                       BGT     : 015A
0147 B6 A04B     K               LDA A     A04B
014A B1 A015                     CMP A     A015
014D 2A 0B      *                BPL     : 015A
014F 5C         \            -   INC B
0150 86 0A                       LDA A #   0A
0152 BB A04C     L               ADD A     A04C
0155 B7 A04C     L               STA A     A04C
0158 20 10                       BRA     : 016A
```

```
S1130100343420 6BFEA048FFA014867108 4A26FCF4
S113011 0FFA016CEA054FFA04EFEA016A60008FF16
S113012 0A016FEA04EA70008813B26EA86068D672E
S113013 0BDE047FFA04AFFA04CFFA0485FB6A04A1D
S113014 0B1A0142B0A2E13B6A04BB1A0152A0B5C38
S113015 0860ABBA04CB7A04C2010860ABBA014B7DB
S113016 0A014860ABBA04AB7A04A7EA0542095206BA
S113017 032FEA014A6008D14FFA014FEA04AA7000E
S113018 0BCA04C270E8D05FFA04A20E55D260209800
S113019 03908398EA0423B20602 0D208080804A5A
S11301A 026F939FEA04886E48DF1FFA018860D8D4E
S11301B 0EAFFA01A86178DE3FFA01C86298DDCFFB9
S11301C 0A01E860D8DD5FFA0286418DCEFFA022D6
S11301D 086018DC78DC5FFA024CE8004FFA028868C
S11301E 001B7A02A86028D11BDE1ACB7A052814DA2
S11301F 027A7814526487EE0D02041860A8D3DFE12
S113020 0A020860C084A26FCBDE07EBDE05516FE03
S113021 0A0140908A60112619FFA04ACEA04A86F8
S113022 0F08D19FEA04A098D108D0E860A8D0DFEE3
S113023 0A04ABCA01626DC20C28D004F2061814458
S113024 027068153269320108607 8D53BDE1AC5FAA
S113025 08159260 15CF7A02C86038D43BDE047FF3E
S113026 0A0148604 8D39BDE047FFA01686058D2FA6
S113027 0BDE0AA81082AF5484816BDE1AC8143Z7B0
S113028 02E8153271C815026E38D0D86FFB7A02AAB
S113029 0A700863FA7012017F7A029FEA028392030
S11302A 0438DF54FB7A02A8613A7008611A700B681
S11302B 0A05281532781 7FA050FEA020BDE07ECEB6
S11302C 0A054BDE1AC810D2708A700088CA07F26AF
S11302D 0F18604A700FEA01CAD00FEA014FFA04AF6
S11302E 08D462018 2065F6A0515C2004860A8D5B9B
S11302F 05A26F98D33FEA01A6E008D088D4F8D7924
S113030 08D1720F67AA05126028D1DFEA04A8D007D
S113031 08D0009BCA0162 7CE39B6A02DFEA04A4AEE
S113032 027F608FFA04A20F786088D1FCEA0548D1B
S113033 069860B8D16860D8D12CEA050A6008B01FA
S113034 019A7008D5E8638A700860920438 60A8D8A
S113035 03FCEA04A86F08D38FEA01EAD00F6A02D3B
S113036 0FEA04A8D168D478D128D108D3FF6A02D5F
S113037 0FEA04A8D0B8D092007 20265A2B2E202300
S113038 05A2B2BA60008 2B26484828224644202016
S113039 04D2B6C276C206E8D1708A600810426F760
S11303A 039207AA6008D48A60008 20478D008620B3
S11303B 0FFA04C847FFEA0287DA02A2B172707BD11
S11303C 0E1D1FEA04C3937E600575724FAA7013390
S11303D 0FEA04C3937A700C637E701C63FE7016DD9
S11303E 0012AFCE60033FEA04C39A60008 20C144D3
S11303F 0444444840F8B30813923B58B0720B18D5D
S1130400A28DA08614FFA04EFEA02037E60008C1EE
S1130410 0426F94A26F68D8233FEA04E39860B8DCA
S1130420 E4FEA018AD0086148DDBFEA02FC6068D59
S1130430 B98DB78DB58DCC8DB18DC8B6A02E2601E2
S1130440 394A2609860C8DBDFEA04A20B24A2609E7
S1130450860E8DB1FEA04A20A84A2622860F8DA5BD
S1130460FEA04AA60008FFA052CEA0524D2A026A5E
S1130470005FAB01E900E700A7012083208 74A263B
S1130480048610 20CD8611F6A02D5A5A27C420B612
S1130490FEA04AFFA03108FFA04A7DA02C26013906
S11304A08D54BCA0262604860C20D18614C680E474
S11304B0 0027024A4A8DF28614C640E40027014AD6
S11304C0 20E720B67DA02C2753FEA01409FFA04AE4
S11304D0FEA04A08FFA04ABCA01626037EE0D0FF77
S11304E0A0318D12BCA02627E78DC0CEA04A8DD2A4
S11304F0860A8DB520DAB6A031843FB7A031FEA0BC
S1130500240907 0808BCA0262601 39A600843FB1A5
S1130510A03126EFA601B1A03226E8397EE0D00052
S1130520 7DA02C260139FEA024FFA0266F006F01B8
S1130530FEA014FFA04AA600817E272981BD272C96
S1130540818D273784F0812027388D78FEA04AB624
S1130550A02D08BCA01627084A26F7FFA04A20D6DB
S1130560 3920E720918680B7A02B20058640B7A0CC
S1130570 2BFEA04AEE01FFA03120248640B7A02B19
S1130580 20058680B7A02B08A60008FFA031CEA0C6
S1130590314D2A026A005FAB01E900E700A7018D33
S11305A0C2BCA026260B6F02086F0208FFA0260912
S11305B009B6A031BAA02BAA00A700B6A032A701A1
S11305C0209F0000FEA04AE600FEA0225C080808 66
S11305D0085A26F9FFA02F8D138D1117270444440C0
S11305E0444C4CB7A02DC407F7A02E398D00A600AB
S11305F048590839000000000D0A0A484541444 59D
S1130600523A20040D0A42595445204445534952 54
S1130610454 43A2004101653575450203638 30308D
S1130620204445534 54D424C4552202056 2E340D0E
S1130630 0A0A0A4F5054494F4E2028442C532C4D3B
S1130640 2C45293A20040D0A0A494E505554204697
S1130650494C4520414444524553533A2004 2054C4
S1130660 4F20040D0A504F5254202620444556 4929
S1130670 4345202830 2D373B20432C532C50293A16
S1130680 20040D0A4E4557204245474 94E4E494ED7
S1130690 47204144445245535 33A20040D0A4D41E6
S11306A0524B20494E5445525345435449 4F4E539F
S11306B0 3F2028592F4E2920040D0A0A0A0A2D2DFD
S11306C0 2D2D0D0A0A0A0A040D0A0A040D0A000453
S11306D0 20202020202020202004202004504147 45B1
S11306E0 2004582C2020043E2004202030300423F1
S11306F0 20042D043E042004000000000000003B
S1130700 3F2020204E4F50203F2020203F202020FB
S1130710 3F2020203F2020205441502054504120 8D
S1130720 494E582044455820434C562053455620A2
S1130730 434C432053454320434C492053454920CF
S1130740 5342412043424120 3F2020203F2020208B
S1130750 3F2020203F2020205441422054424120 69
S1130760 3F2020244414120 3F2020204142420 7D
S1130770 3F2020203F2020203F2020203F202020F9
S1130780 4252C1A03F2020204248C9A0424CD3A0DD
S1130790 4243C3A04243D3A0424EC5A04245D1A088
S11307A0 4256C3A04256D3A04250CCA0424DC9A049
S11307B0 4247C5A0424CD4A04247D4A0424CC5A055
S11307C0 5453535820494E532050554C4150554C4297
S11307D0 4445532054585320505348415053484 2A1
S11307E0 3F2020205254532 03F2020205254920 9F
S11307F0 3F2020203F2020205741492053574920A3
S1130800 4E4547413F2020203F202020434F4D416B
S1130810 4C5352413F202020524F52414153524 1A8
S1130820 41534C41524F4C41444543413F202020C9
S1130830 494E434154535441 3F202020434C52419C
S1130840 4E4547423F2020203F202020434F4D4229
S1130850 4C5352423F202020524F52421535242 65
S1130860 41534C42524F4C42444543423F20202086
```

```
S11308704946434254535442 3F20202043 4C524259
S11308804E45C7203F20202033F20202043 4FCD202D
S11308904C53D2203F20202052 4FD220415 3D2200B
S11308A04153CC20524FCC20445C3203F20202 02C
S11308B0494EC3205453D4204A4DD020434CD22017
S11308C0CE4547A03F20202033F20202 0C34F4DA0ED
S11308D0CC5352A03F20202 0D24F52A0C15352A04B
S11308E0C1534CA0D24F4CA0C44543A03F20202 06C
S11308F0C94E43A0D45354A0CA4D50A0C34C52A0D7
S1130900D53D542C143CD50C153C243C13F20202 0DF
S1130910 41CE44C142C954C14CC441C13F202020EE
S1130920 45CF52C141C443C14FD241C141C444C166
S1130930C3D058A04253D2A0CCC453A03F202020FF
S1130940 53D5424143CD504153C243413F20202 01F
S1130950 41CE444142C954414CC4414153D4414124
S1130960 45CF524141C443414FD2414141C4444126
S1130970 43D058203F2020204CC4532053D453202C
S1130980 5855C241434DD0415342C3413F20202 0DF
S1130990414EC4414249D4414C44C1415354C141E4
S11309A0454FD2414144C3414F52C1414144C441E6
S11309B04350D8204A53D2204C44D3205354D320FC
S11309C0D35542C1C34D50C1D34243C13F20202 01F
S11309D0C14E44C1C24954C1CC4441C1D35441C1A4
S11309E0C54F52C1C14443C1CF5241C1C14444C1A6
S11309F0C35058A0CA5352A0CC4453A0D35453A0BC
S1130A0053D5420243CD50C253C243C23F20202 0DB
S1130A1041CE44C2C24254C24CC441C23F202020F1
S1130A2045CF52C241C443C24FD241C241C444C261
S1130A303F2020203F20202 0CCC458A03F20202 04D
S1130A4053D5424243CD504253C243423F20202 01B
S1130A5041CE444242C954424CC4414253D441421F
S1130A6045CF524241C443424FD2414241C4444221
S1130A703F2020203F20202 04CC4582053D458200D
S1130A805355C242434DD0425342C3423F20202 0DB
S1130A90414EC4424249D4424C44C1425354C142DF
S1130AA0454FD2424144C3424F52C1424144C442E1
S1130AB03F2020203F20202 04C44D8205354D820CD
S1130AC0D35542C2C34D50C2D34243C23F20202 01B
S1130AD0C14E44C2C24954C2CC4441C2D35441C29F
S1130AE0C54F52C2C14443C2CF5241C2C14444C2A1
S1130AF03F2020203F20202 0CC4458A0D35458A08D
```

NOTICE TO USERS OF SWTPC PAPER AND CASSETTE TAPES

In order to help reduce the time necessary to load programs through either a paper tape reader or an SWTPC AC-30 cassette interface, the longer tapes supplied from SWTPC will be furnished in a binary format instead of the conventional ASCII. At the beginning of each tape is a binary loader program that will load into the computer using the regular ASCII format. The program then executes itself and loads the main program in binary. Using this method, tapes will load in approximately 1/3 normal time. When using an SWTPC AC-30, **lock** the reader in the ON position and type L. For paper tapes readers, such as on an ASR-33 Teletype,® when the load stops after the binary loader has been loaded into the computer simply type G. This will execute the binary loader and the remainder of the tape will load into memory. Several "garbage" characters may be printed immediately after the binary loader loads in—this is normal. On cassette tapes, one side will be in conventional ASCII (side with long leader) and one side will be in binary. The tapes are formatted as follows:

L	BINARY LOADER IN ASCII	S9	G	MAIN PROGRAM IN BINARY

As the tape loads, you will see one of the following displays on your terminal: (either is OK)

*L	*L
*G	*??
* (register dump)	* (register dump)

Some tapes may have an additional feature which will verify that the tape loaded correctly into memory. If, after loading the tape, you find that the program counter is not automatically set to the correct value then you probably have a verifying tape. If this is the case simply typing a G will automatically check the validity of the program and execute it. If the message LOAD ERROR is displayed then the tape did not load correctly into memory. The most common cause of this is a memory problem—there can be problems that MEMCON and ROBIT will not find.

The format for a self-verifying tape is as follows:

VERIFICATION ROUTINE	BINARY LOADER PGM. CTR.	BINARY LOADER IN ASCII	S9	G	MAIN PROGRAM IN BINARY	MAIN PROGRAM PGM. CTR.
ASCII					BINARY	

As before, one side of a cassette tape will be in binary and the other side in ASCII.

If you are unable to load a tape please check the following:

1.) Be sure the reader is locked on to load a binary cassette tape.
2.) Try different volume and tone control settings.
3.) Clean your tape heads with alcohol and a cotton swab.
4.) Re-check all memory if a LOAD ERROR is displayed.

Teletype® is a registered trademark of Teletype Corp.